Self-Printing Programs

Gene Michael Stover

created 2006 March 1
updated Friday, 2006 May 26

1 What is this?

Someone sent an e-mail to a coworker, saying that a really difficult, almost impossible programming task was to write a program which would print its own source code.

Yes, yes yes, I recognize that all of this has been done before. But it hasn't been done before by me.

2 First Solution

I thought about it for a minute, then wrote a first C version[1] & a first MS-DOG `.bat` version[2] Those programs both use the same trick to satisfy the problem. On the one hand, I feel that these two programs satisfy the programming task, but I also admit that they use a dirty trick to do it because they require a data file at run-time.

I also wrote a first Lisp version[3] It isn't portable; it requires *clisp* to run. I also could have use the "dirty trick" which I used in the C version.

3 Second Implementation

Days later, I realized that you could do it if the program has an incomplete string representation of its own source code, prints that string & also inserts that string into itself in a key location. That gives you source code which compiles to a program which again prints the incomplete string & substitutes the string into itself, giving you the souce code for that program.

It was totally straight-forward in Lisp. (See Appendix D.)

[1] For the first C version, see Appendix A.

[2] For the first MS-DOG version, see Appendix B.

[3] For the first Lisp version, see Appendix C.

It was more difficult in C because C's `printf` does not contain a verbatim substituion field like Lisp's FORMAT's `~S`. So I wrote a C program which produces a program which prints its own source code. In other words, the C program, the first time it's run, reads its source code from a file. It produces the source code for a program which prints its own source code by using the "substitute the incomplete string into itself" technique. The C program I wrote is in Appendix E. The C program it produces is in Appendix F.

4 What is a self-printing program

The programming problem was presented to me as "Can you write a program which prints its *exact* source code?"

That got the point across, but I suspect a more precise definition of a self-printing program is...

1. source code *S* which compiles to an executable program *P*,
2. when executed, *P* produces output *T*, &
3. *T* is equivalent to *S*. (If *T* & *S* are byte-for-byte identical, then it's a no-brainer that they are equivalent.)

A First C Implementation

This is the first C implementation. It uses a "dirty trick" in that it requires the presence of the source code file at run-time.

This source code is also at `http://cybertiggyr.com/gene/aaa/showself.c`.

```
/*
 * $Header: /home/gene/library/website/docsrc/aaa/RCS/showself.c,v 395.1 2008/04/20 17:25:45 gene
 */

#include <stdio.h>

int
main ()
{
  FILE *fp;
  int c;

  fp = fopen ("showself.c", "r");
  while ((c = fgetc (fp)) != EOF) {
    putchar (c);
  }
  return 0;
}
```

B First MS-DOG .BAT Implementation

This is the first MS-DOG `.bat` implementation. It uses a “dirty trick” in that it requires the presence of the source code file at run-time.

This source code is also at `http://cybertiggyr.com/gene/aaa/showself0.bat`.

```
@echo off
REM This is SHOWSELF0.BAT
type showself0.bat
REM --- end of file ---
```

C First Lisp Implementation

This is the first Lisp implementation. It requires a non-portable feature of *clisp*. Other Lisps surely have a similar feature in their own non-portable ways, so you could use this same trick for them.

This source code is also at `http://cybertiggyr.com/gene/aaa/showself.lisp`.

```
(defun showself ()
  (first (second (SYMBOL-PLIST ’showself))))
```

D Second Lisp Implementation

Here is a Lisp program which, in my opinion, satisfies the programming task without using any dirty tricks. It is a self-contained program which prints its own source code.

This source code is also at `http://cybertiggyr.com/gene/aaa/showself1.lisp`.

```
;;; File: showself1.lisp

(defvar *src* ";;; File: showself1.lisp

(defvar *src* ~S)

(defun showself1 ()
  (format t *src* *src*))
")

(defun showself1 ()
  (format t *src* *src*))
```

E Second C Implementation, part A

Here is a C program which reads its source code from a file & prints a program which prints its own source code *even when the file is not present.* This program’s output, which is the program in in Appendix F, is a program which prints its own source code.

This source code is also at http://cybertiggyr.com/gene/aaa/showself1.c.

```
/* File: showself1.c */

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

static char S_src[] = "%ooga%";

static void
S_PrintSub (char *from, char *to)
{
  char *p;

  for (p = from; p < to; ++p) printf ("%c", *p);
}

static void
S_PrintData (char str[])
{
  int i = 0;

  while (str[i] != '\0') {
    switch (str[i]) {
    case '"': printf ("\\\""); break;
    case '\\': printf ("\\\\"); break;
    case '\n': printf ("\\n\"\n  \""); break;
    default: putchar (str[i]);
    }
    ++i;
  }
}

static void
S_PrintRest (char str[])
{
  char *p;

  for (p = str; *p != '\0'; ++p) {
    putchar (*p);
  }
}

int
main ()
{
  char *src, *p;
  FILE *fp;
  int i;
  if (strcmp (S_src, "%ooga%") == 0) {
    /* Must load the source code from the file. */
    fp = fopen ("showself1.c", "r");
    src = (char *) malloc (10 * 1024);
    fread (src, 1, 10 * 1024, fp);
    fclose (fp);
  } else {
    src = S_src;
  }
  p = strstr (src, "%ooga%");
  i = strlen ("%ooga%");
  S_PrintSub (src, p);
  S_PrintData (src);
  S_PrintRest (p + i);
  return 0;
}
```

F Second C Implementation, part B

This program was produced by the program in Appendix E. This program is a program which prints its own source code. Unlike my first solutions to the programming task, this one works even if the source code file is not available at run-time.

This source code is also at http://cybertiggyr.com/gene/aaa/showself2.c.

```
/* File: showself1.c */

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

static char S_src[] = "/* File: showself1.c */\n"
  "\n"
  "#include <stdlib.h>\n"
  "#include <stdio.h>\n"
  "#include <string.h>\n"
  "\n"
  "static char S_src[] = \"%ooga%\";\n"
  "\n"
  "static void\n"
  "S_PrintSub (char *from, char *to)\n"
  "{\n"
  "  char *p;\n"
  "\n"
  "  for (p = from; p < to; ++p) printf (\"%c\", *p);\n"
  "}\n"
  "\n"
  "static void\n"
```

```
"S_PrintData (char str[])\n"
"{\n"
"  int i = 0;\n"
"\n"
"  while (str[i] != '\\0') {\n"
"    switch (str[i]) {\n"
"    case '\"': printf (\"\\\\\\\"\"); break;\n"
"    case '\\\\': printf (\"\\\\\\\\\\\"); break;\n"
"    case '\\n': printf (\"\\\\n\\\"\\n  \\\"\"); break;\n"
"    default: putchar (str[i]);\n"
"    }\n"
"    ++i;\n"
"  }\n"
"}\n"
"\n"
"static void\n"
"S_PrintRest (char str[])\n"
"{\n"
"  char *p;\n"
"\n"
"  for (p = str; *p != '\\0'; ++p) {\n"
"    putchar (*p);\n"
"  }\n"
"}\n"
"\n"
"int\n"
"main ()\n"
"{\n"
"  char *src, *p;\n"
"  FILE *fp;\n"
"  int i;\n"
"\n"
"  if (strcmp (S_src, \"%ooga%\") == 0) {\n"
"    /* Must load the source code from the file. */\n"
"    fp = fopen (\"showself1.c\", \"r\");\n"
"    src = (char *) malloc (10 * 1024);\n"
"    fread (src, 1, 10 * 1024, fp);\n"
"    fclose (fp);\n"
"  } else {\n"
"    src = S_src;\n"
"  }\n"
"  p = strstr (src, \"%ooga%\");\n"
"  i = strlen (\"%ooga%\");\n"
"  S_PrintSub (src, p);\n"
"  S_PrintData (src);\n"
"  S_PrintRest (p + i);\n"
"  return 0;\n"
"}\n"
"";
```

```
static void
S_PrintSub (char *from, char *to)
{
  char *p;

  for (p = from; p < to; ++p) printf ("%c", *p);
}

static void
S_PrintData (char str[])
{
  int i = 0;

  while (str[i] != ’\0’) {
    switch (str[i]) {
    case ’"’: printf ("\\\""); break;
    case ’\\’: printf ("\\\\"); break;
    case ’\n’: printf ("\\n\"\n  \""); break;
    default: putchar (str[i]);
    }
    ++i;
  }
}

static void
S_PrintRest (char str[])
{
  char *p;

  for (p = str; *p != ’\0’; ++p) {
    putchar (*p);
  }
}

int
main ()
{
  char *src, *p;
  FILE *fp;
  int i;

  if (strcmp (S_src, "%ooga%") == 0) {
    /* Must load the source code from the file. */
    fp = fopen ("showself1.c", "r");
    src = (char *) malloc (10 * 1024);
    fread (src, 1, 10 * 1024, fp);
    fclose (fp);
  } else {
    src = S_src;
  }
  p = strstr (src, "%ooga%");
  i = strlen ("%ooga%");
  S_PrintSub (src, p);
  S_PrintData (src);
  S_PrintRest (p + i);
  return 0;
}
```

References